Параллельные алгоритмы IBM Research для решения задач обхода и построения кратчайших путей на графах с триллионами ребер

Александр Позднеев
Технический консультант по высокопроизводительным системам
IBM Восточная Европа/Азия

Семинар “GraphHPC-2014”, 5 марта 2014 г.

Содержание

- Введение
- Задача поиска в ширину
  - IBM Blue Gene в Graph500
  - Структуры данных и алгоритмы
  - Анализ производительности
- Задача поиска кратчайших путей
  - Структуры данных и алгоритмы
  - Анализ производительности
- Заключение

Введение

Эпоха Big Data

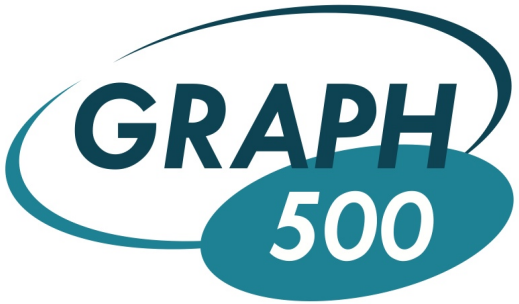
- Огромные массивы данных
- Нет локальности данных, под которую оптимизированы существующие архитектуры
- Нерегулярный доступ – кэш и предвыборка не работают
- Низкое соотношение объема вычислений и числа операций доступа к памяти – не скрыть латентность
- Маленькие сетевые пакеты в случайных направлениях с высокой частотой

Задача поиска в ширину (Breadth-First Search, BFS)

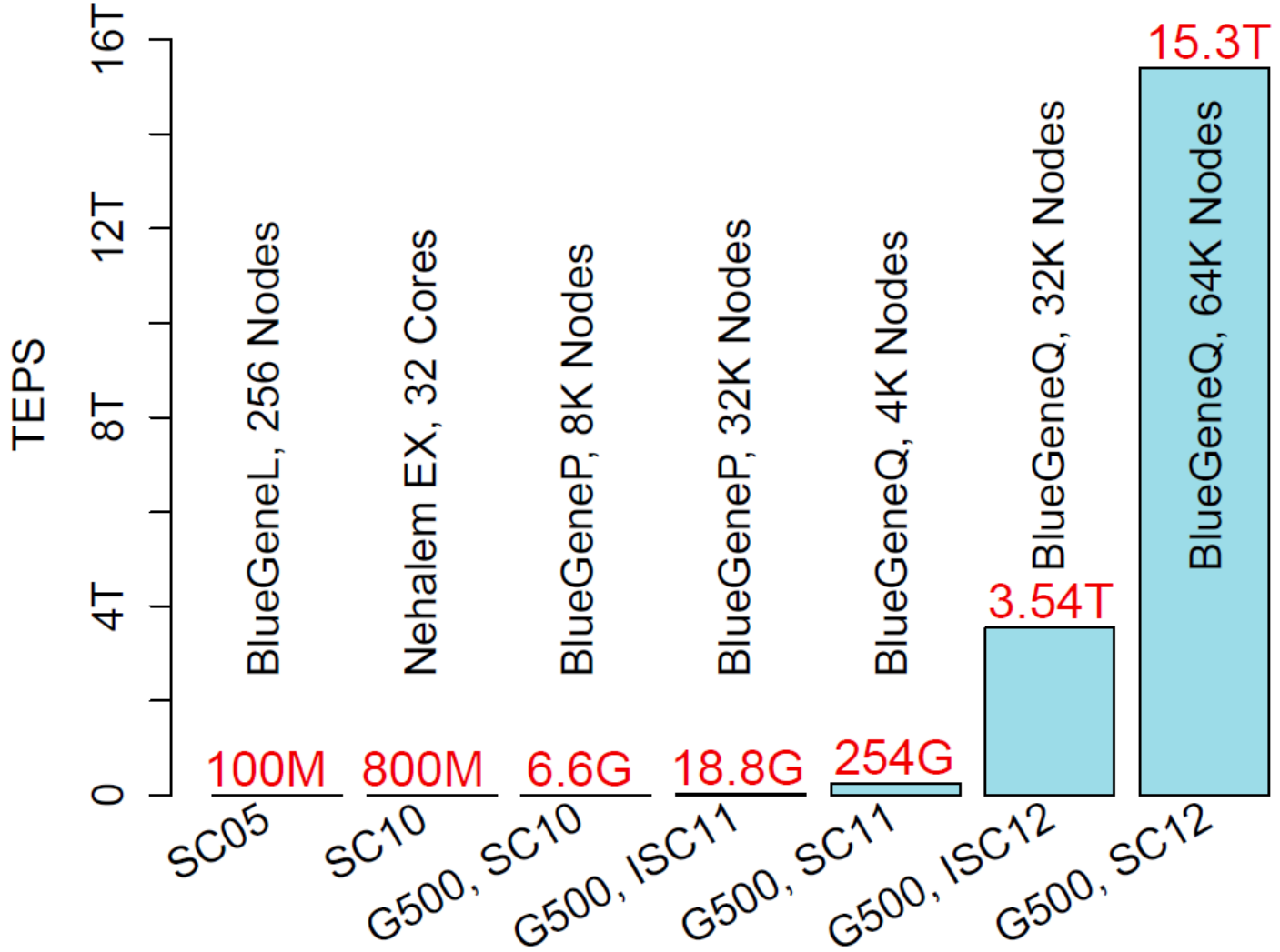
F. Checconi, F. Petrini “Traversing Trillions of Edges in Real-time: Graph Exploration on Large-scale Parallel Machines”, IPDPS 2014, Phoenix, Arizona, May 19-23, 2014
http://www.odbms.org/wp-content/uploads/2014/05/g500-ipdps14.pdf

Blue Gene в Graph500

Дата	Место	Система	Модель	Число ядер
2013, ноябрь	1	Sequoia	Q	1 048 576
2013, июнь	1	Sequoia	Q	1 048 576
2012, ноябрь	1	Sequoia	Q	1 048 576
2012, июнь	1	Sequoia / Mira	Q	524 288
2011, ноябрь	1	NNSA/SC (IBM – BG/Q Prototype)	Q	65 536
2011, июнь	1	Interpid	P	131 072
2010, ноябрь	1	Interpid	P	32 000



BFS: Эволюция производительности

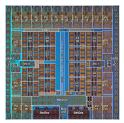


BFS: Рекордные результаты

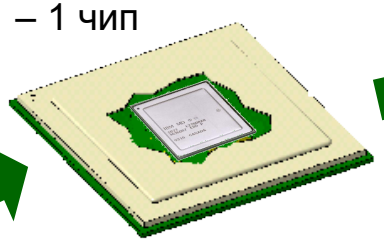
- IBM BG/Q Sequoia
  - 64 стойки (из 96)
  - 64k выч. узлов
  - 4 млн. нитей
- R-MAT
  - scale=40
  - 1 трлн. вершин ~10^{12}
- **15.3 трлн. TEPS**

BG/Q

1. Чип
– 16+2 ядер



2. Модуль
– 1 чип



3. Вычислительный узел
– 1 чиповый модуль
– 16 ГБ памяти DDR3

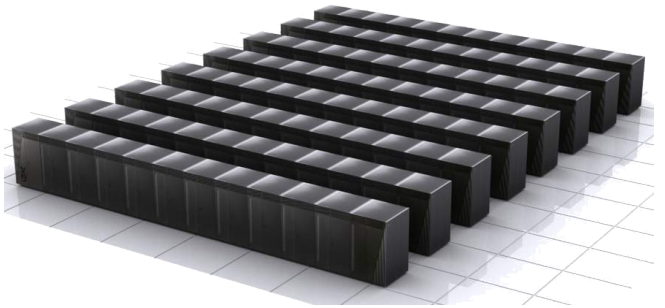
4. Плата вычислительных узлов
– 32 вычислительных узла
– оптические модули
– чипы коммуникаций
– сеть 5D-тор

5a. Шасси вычислительных узлов
– 16 плат вычислительных узлов

5b. Шасси узлов ввода-вывода
– 8 узлов ввода-вывода
– 8 слотов PCIe Gen2 x8

6. Стойка
– 2 шасси вычислительных узлов
– 0, 1, 2 или 4 шасси узлов ввода-вывода

7. Система из многих стоек

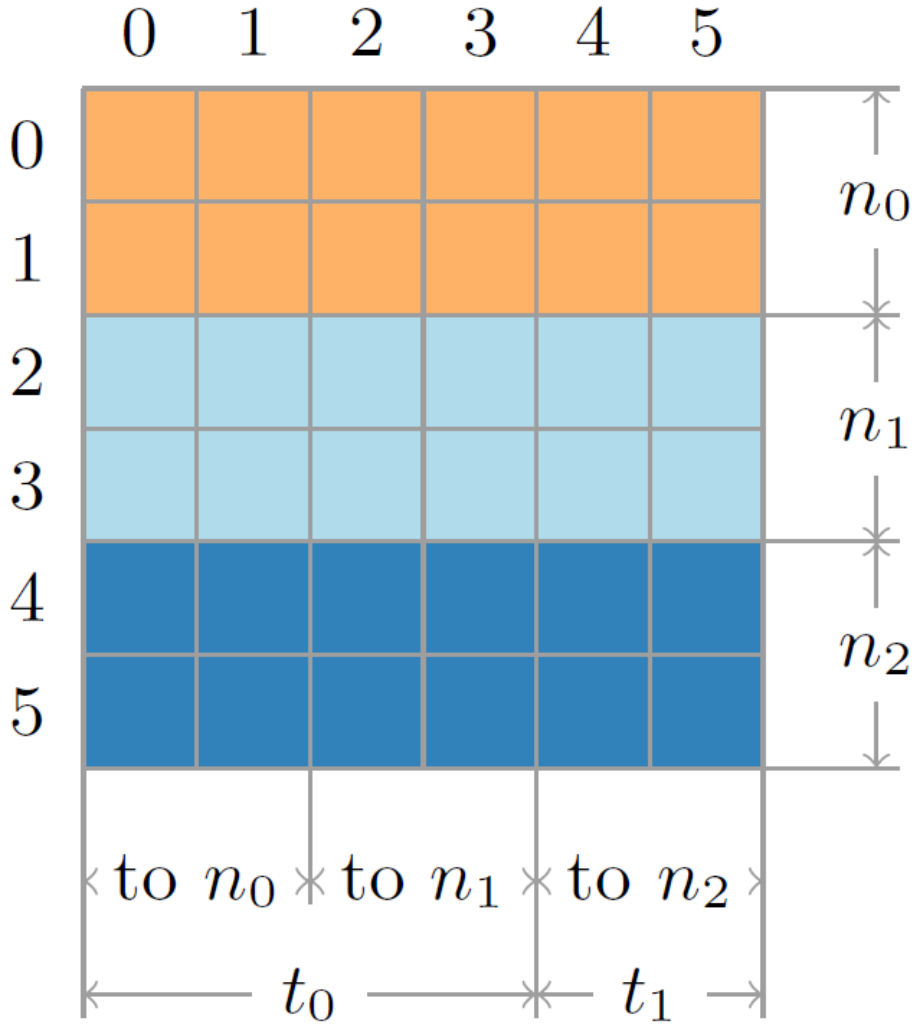


BFS: Последовательный алгоритм

```
Input: G = (V, E): graph representation;
       v_s: source vertex;
       In: current level input vertices;
       Out: current level output vertices;
       Vis: vertices already visited.
Output: P: predecessor map.
1  In ← {v_s} ;
2  Vis ← {v_s} ;
3  P(v) ← ⊥ ∀v ∈ V ; P(v_s) ← v_s
4  while In ≠ ∅ do
5      // Find the reachable edges.
6      Out ← ∅ ;
7      for u ∈ In do
8          for v | (u, v) ∈ E do
9              if v ∉ Vis then
10                 Out ← Out ∪ {v} ;
11                 Vis ← Vis ∪ {v} ;
12                 P(v) ← u ;
13     // Prepare for next level.
14     In ← Out ;
```

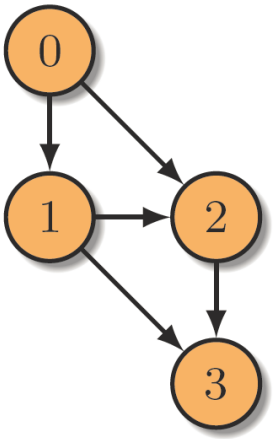
BFS: Одномерная декомпозиция

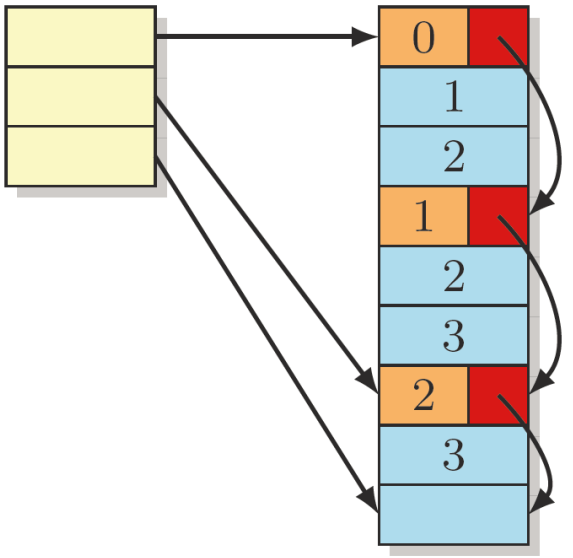
- Вершины поделены между *m* выч. узлами: n_0, n_1, …, n_{m-1}
- Ребра, инцидентные вершинам выч. узла n_i, хранятся на выч. узле n_i
- Распределенный алгоритм: ребра генерируют сообщения от узла-владельца src-вершины узлу-владельцу dst-вершины



BFS: Compressed adjacency list + a coarse index

- [(вершина, ссылка), [соседи]+]+
  - (вершина, ссылка)
    - старшие биты – из номера выч. узла
    - сэкономленные биты – ссылка (# соседей)
  - [соседи]+
    - нужны все 40 бит
    - сэкономленные – для балансировки нагрузки





- coarse index
  - каждые 64 вершины
  - пропускать большие части списка на начальных и конечных шагах, проверяя слова битового массива *In*

BFS: Search pruning – direction-оптимизация

- Граф малого диаметра – на некоторых шагах можно пройти на порядок меньше ребер
- Вместо top-down – bottom-up
  - каждая еще недостигнутая вершина проверяет, не принадлежит ли ее сосед текущему фронту
  - одно или два сообщения на ребро:
    - протестировать потенциального родителя
    - заявить родительские права

BFS: Direction-оптимизированный алгоритм

Input: $G = (V, E)$: graph representation;
n: rank;
v_s: source vertex;
In: current level input vertices;
Out: current level output vertices.
Output: P: predecessor map.

1 $In_r \leftarrow \begin{cases} \{v_s\} & \text{if } v_s \in R_r \\ \emptyset & \text{otherwise} \end{cases}$;
2 $Vis_r \leftarrow \{v_s\}$;
3 $P(v) \leftarrow \perp \forall v \in V$;
4 **if** $v_s \in R_r$ **then** $P(v_s) \leftarrow v_s$;
5 **while** $In \neq \emptyset$ **do**
6 $dir \leftarrow$ CalcDirection() ;
7 **if** $dir = FORWARD$ **then**
8 ForwardStep() ;
9 **else**
10 BackwardStep() ;
11 $In \leftarrow Out$;

CalcDirection():

$NV_{\textit{not visited}}$

$NE_{\textit{reachable from frontier}}$

```
if (NV > NE)
  dir = FORWARD
else
  dir = BACKWARD
```

BFS: ForwardStep(), BackwardStep()

```
Input: G = (V, E): graph representation;
       n: rank;
       In: current level input vertices;
       Out: current level output vertices;
       Vis: vertices already visited.
Output: P: predecessor map.
1  function ForwardStep ()
2  begin
3      for u ∈ In_n do
4          for v : (u, v) ∈ E_n do
5              send (u, v, FORWARD) to Owner (v) ;
6  function BackwardStep ()
7  begin
8      for v ∈ ¬Vis_n do
9          for u : (u, v) ∈ E_n do
10             send (u, v, BACKWARD) to Owner (u) ;
11 function Receive (u, v, dir)
12 begin
13     if dir = FORWARD then
14         if v ∉ Vis_n then
15             Vis_n ← Vis_n ∪ {v} ;
16             Out_n ← Out_n ∪ {v} ;
17             P(v) ← u ;
18     else
19         // Backward.
20         if u ∈ In_n then
21             send (u, v, FORWARD) to Owner (v) ;
```

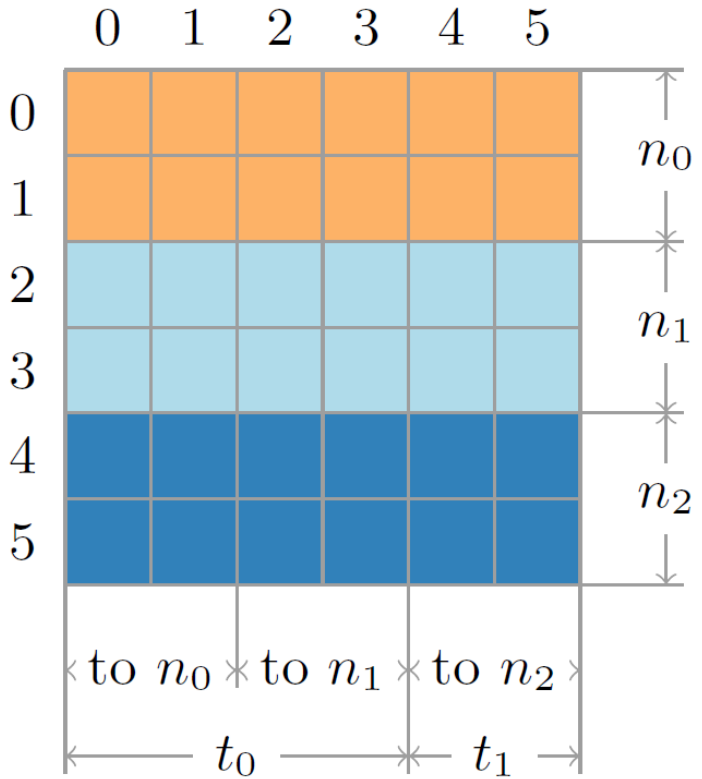
- ForwardStep()
  - top-down
  - одно сообщение на ребро узлу-владельцу посещенной вершины
- BackwardStep()
  - bottom-up
  - одно сообщение на ребро потенциальному родителю

BFS: Баланс вычислений и обменов

- Эффективность утилизации вычислительных и коммуникационных ресурсов
- Полностью асинхронные коммуникации
- Полное перекрытие вычислений и обменов
- Баланс между вычислениями и обменами
  - в больших системах лимитирующий фактор – bisection bandwidth
  - неизрасходнованная вычислительная мощность – на сжатие сообщений

BFS: Коммуникации

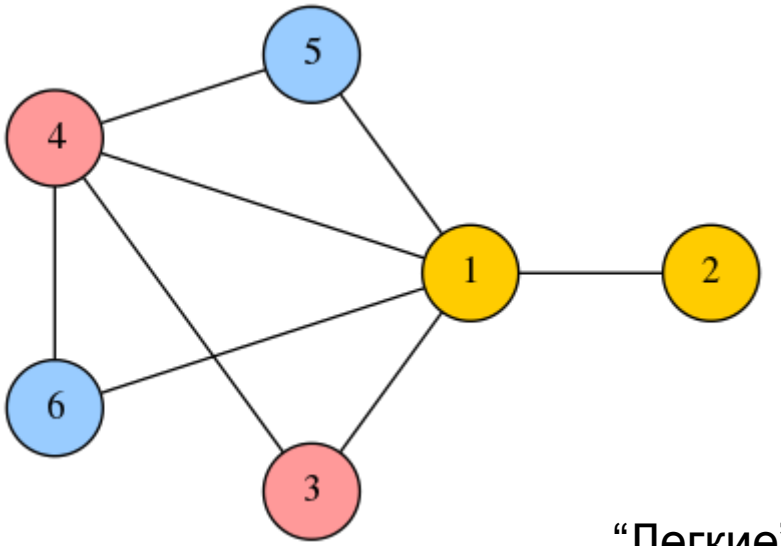
- В каждом выч. узле для каждого выч. узла поддерживается буфер сообщений
- Список ребер каждой вершины разделен между нитями узла – различным нитям всегда соответствуют различные узлы
- Каждый блок $A_{i,j}$ матрицы связности приписан определенной нити, и нити управляются с сообщениями независимо



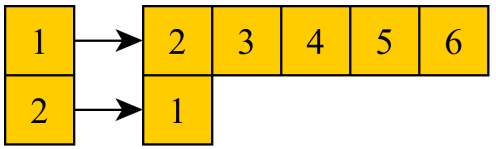
BFS: Балансировка нагрузки

- Причины
  - нерегулярная структура графов
  - наличие вершин с огромным числом соседей
- *LB* – число вершин, подвергаемых балансировке (**L**oad **B**alancing)
- Множество *H* (**H**eavy) – вершины (в количестве *LB*) с самыми большими степенями
  - формируется на этапе распределения данных
  - не входят в одномерную декомпозицию

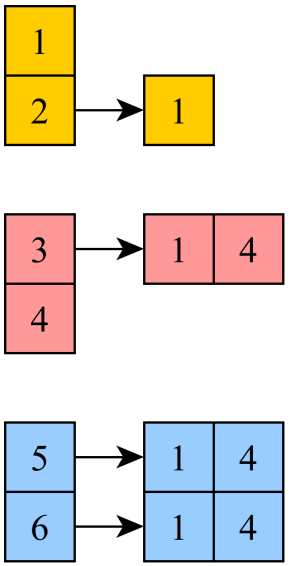
BFS: Схема распределения данных



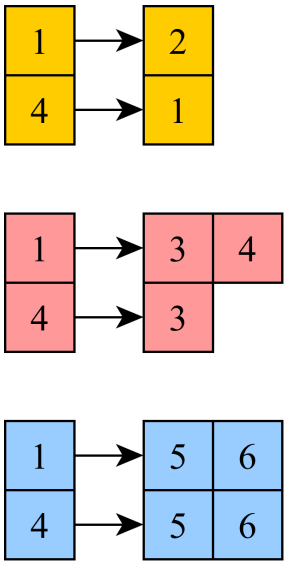
Классическая одномерная декомпозиция



“Легкие” вершины



“Тяжелые” вершины



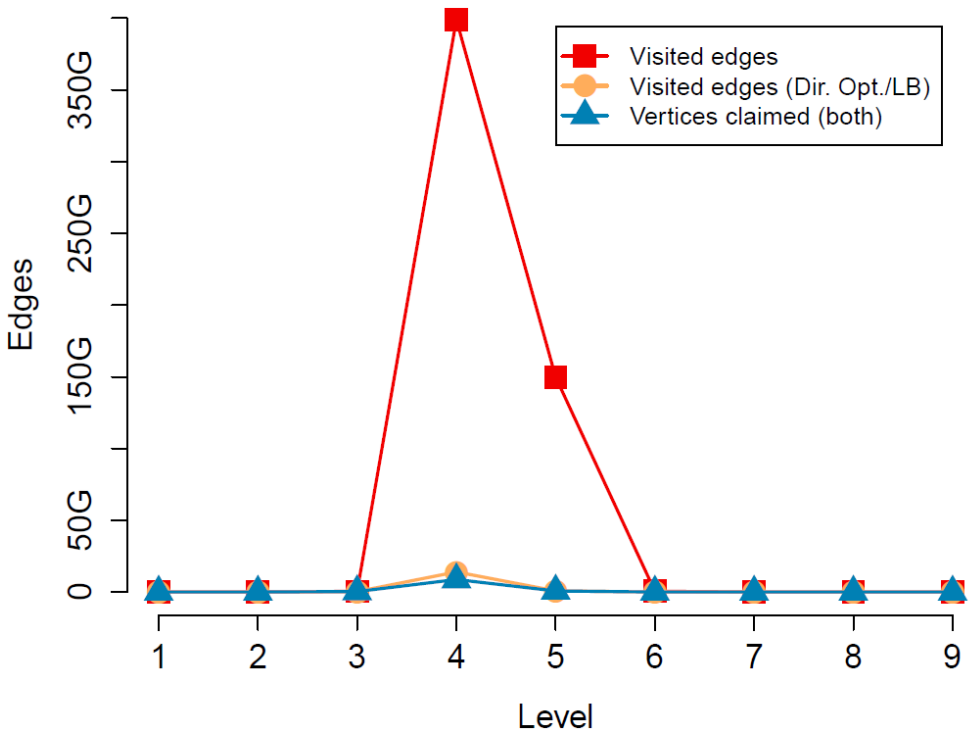
BFS: Алгоритмы для H-вершин

```
1  function ForwardStep ()
2  begin
3      Out_n^H ← ∅ ;
4      for u ∈ In^H do
5          for v : (u, v) ∈ E_n do
6              if v ∈ H then
7                  Vis_n^H ← Vis_n^H ∪ {v} ;
8                  Out_n^H ← Out_n^H ∪ {v} ;
9                  P_n^H(v) ← u ;
10             else
11                 Vis_n ← Vis_n ∪ {v} ;
12                 Out_n ← Out_n ∪ {v} ;
13                 P(v) ← u ;
14     allreduce (Vis_n^H, OR) ;
15     allreduce (Out_n^H, OR) ;
16     In^H ← Out_n^H ;
```

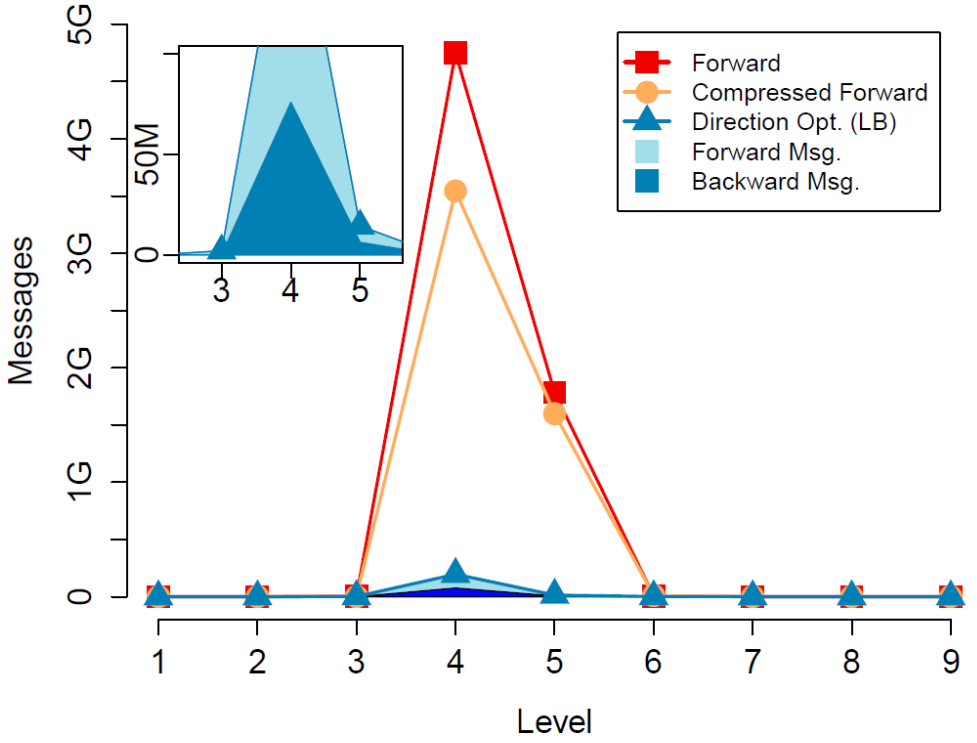
```
17 function BackwardStep ()
18 begin
19     Out_n^H ← ∅ ;
20     for v ∈ ¬Vis_n^H do
21         for u : (u, v) ∈ E_n do
22             if u ∈ H then
23                 if u ∈ In_n^H then
24                     Vis_n^H ← Vis_n^H ∪ {v} ;
25                     Out_n^H ← Out_n^H ∪ {v} ;
26                     P_n^H(v) ← u ;
27                 else if u ∈ In_n then
28                     Vis_n ← Vis_n ∪ {v} ;
29                     Out_n ← Out_n ∪ {v} ;
30                     P(v) ← u ;
31     allreduce (Vis_n^H, OR) ;
32     allreduce (Out_n^H, OR) ;
33     In^H ← Out_n^H ;
```

BFS: Эффект direction-оптимизации

Свойства графа

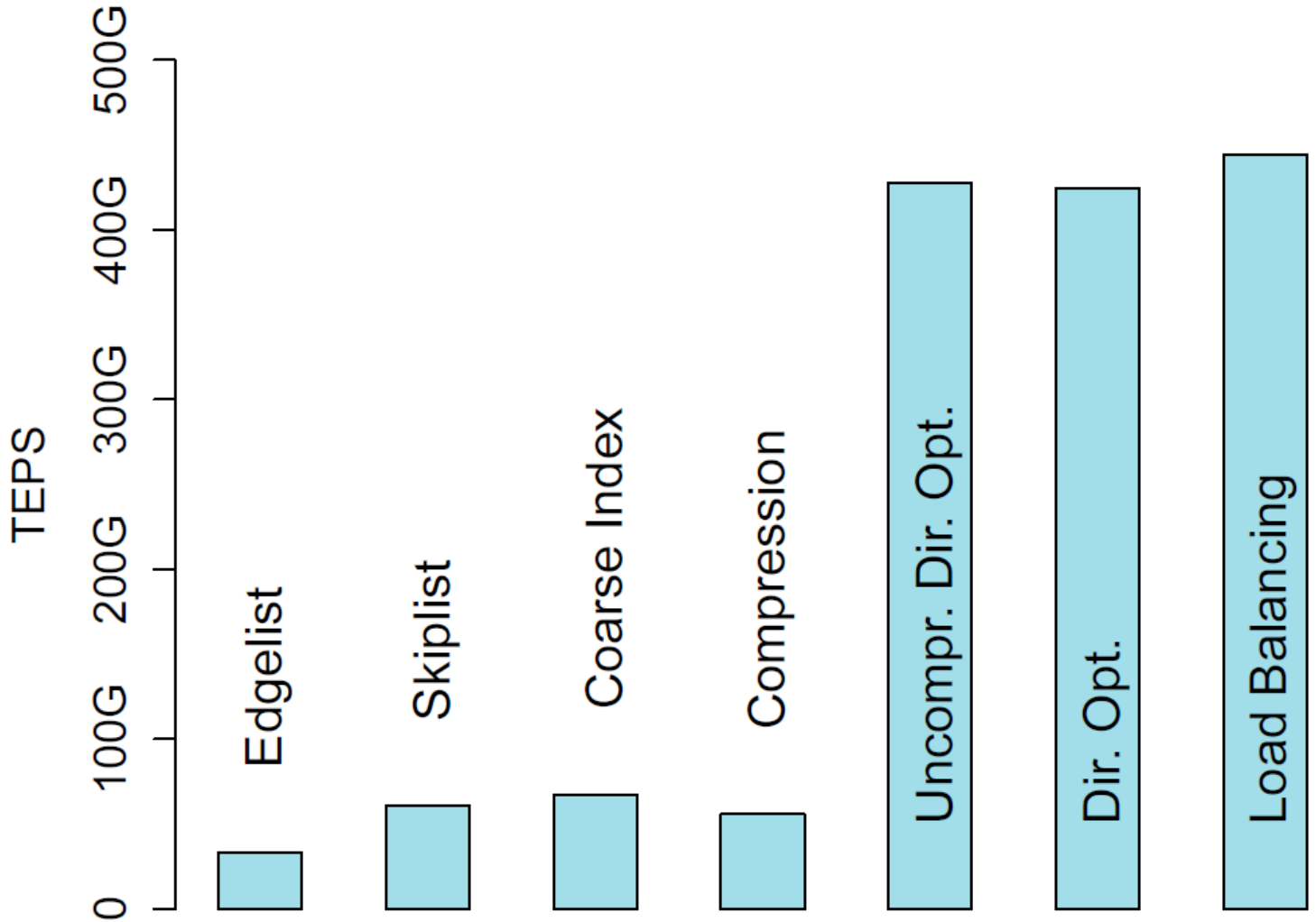


Число сообщений



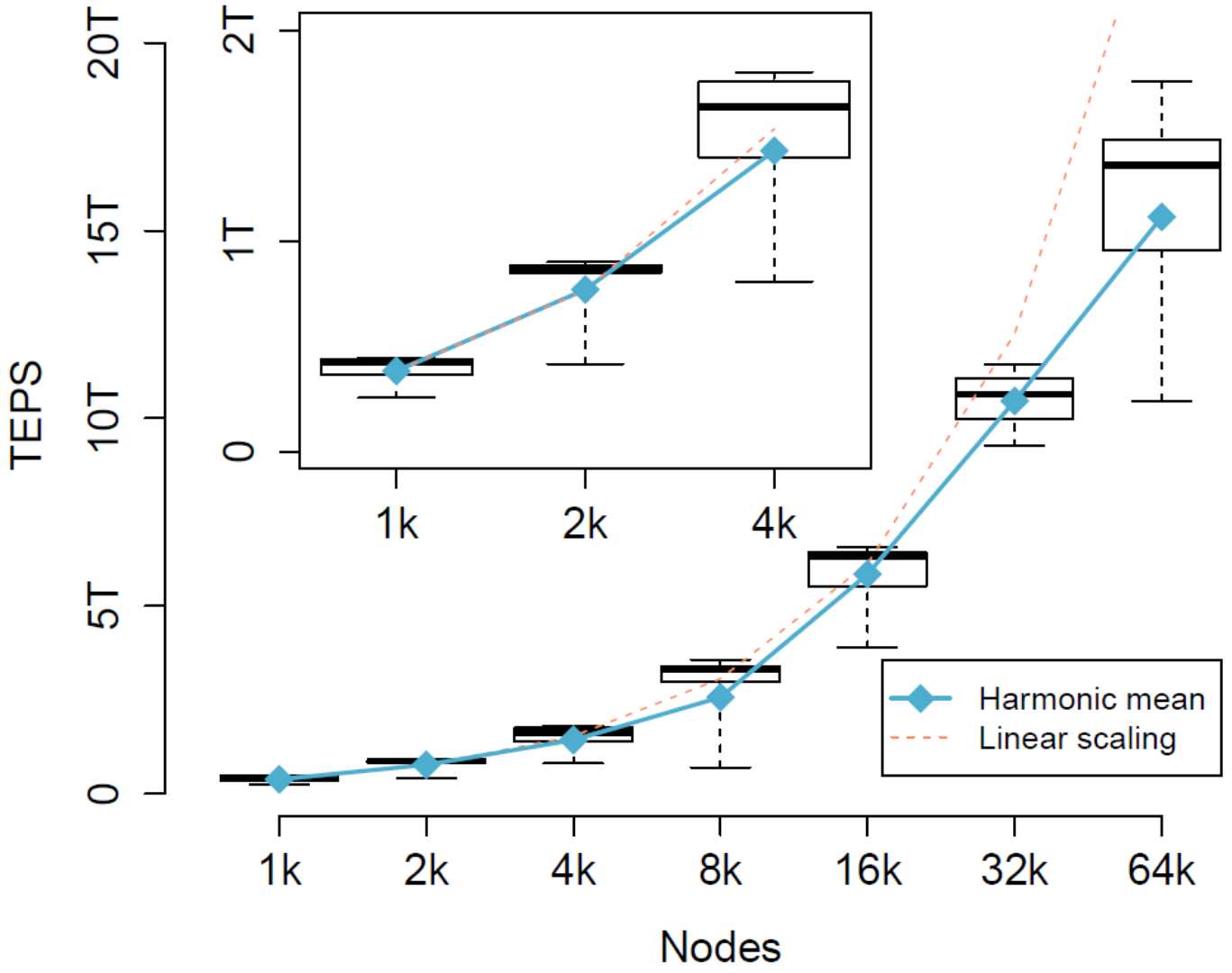
1024 узла, scale=34

BFS: Эффект отдельных оптимизаций



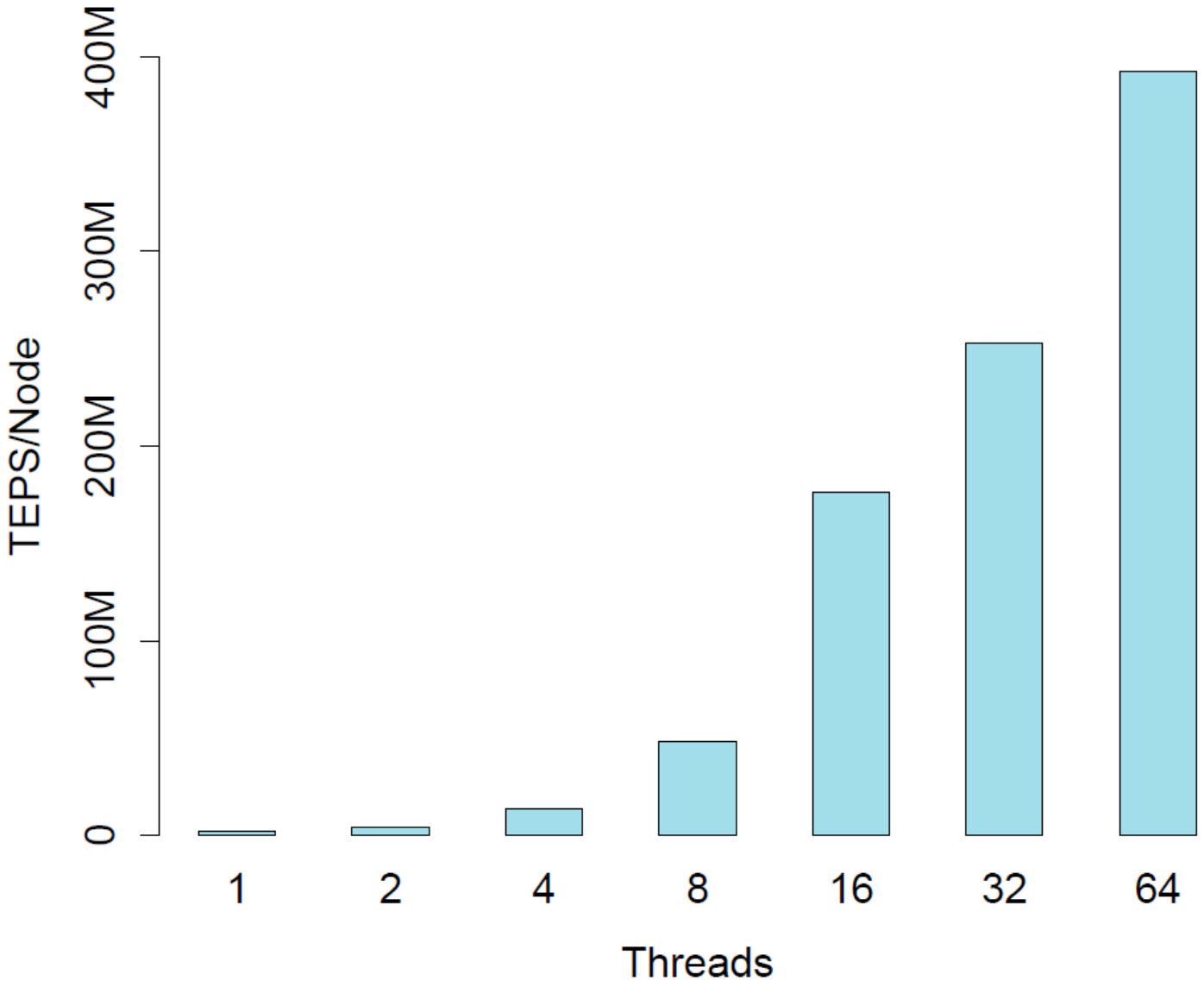
1024 узла, scale=34

BFS: Масштабируемость в слабом смысле



scale: 34-40

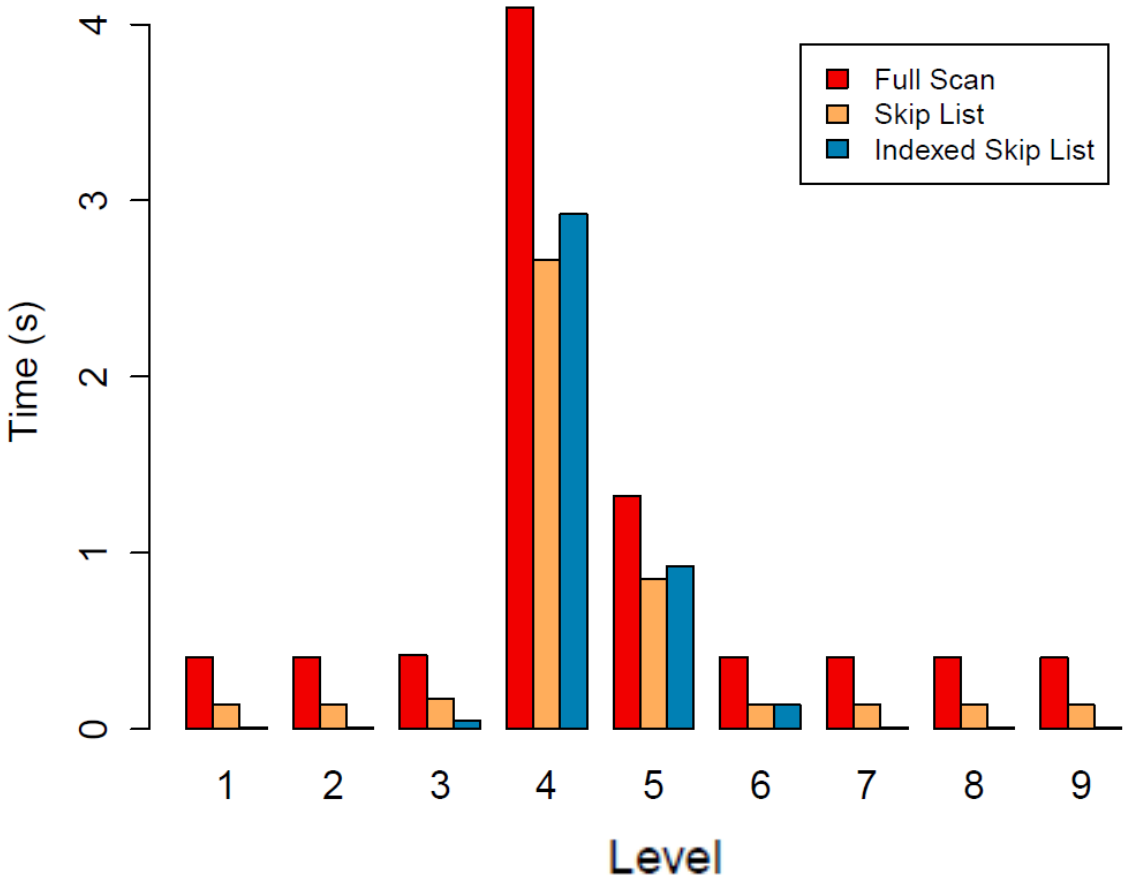
BFS: Эффект многопоточности



1024 узла, scale=34

BFS: Влияние представления графа

- 1024 узла, scale=34
- Full Scan – пары вершин
- Skip List – Compressed Adjacency list
- Indexed Skip List – Skip List + Coarse Index
- ~~Direction-оптимизация~~



Задача поиска кратчайших путей (Single-Source Shortest Path, SSSP)

V.T. Chakaravarthy, F. Checconi, F. Petrini, Y. Sabharwal “Scalable Single Source Shortest Path Algorithms for Massively Parallel Systems”, IPDPS 2014, Phoenix, Arizona, May 19-23, 2014
http://www.odbms.org/wp-content/uploads/2014/05/sssp-ipdps2014.pdf

SSSP: Рекордные результаты

- IBM BG/Q Mira
  - 32 стойки (из 48)
  - 32k выч. узлов
  - 2 млн. нитей
- R-MAT
  - scale=38
  - 0.25 трлн. вершин
- **3.1 трлн. TEPS**

BFS и SSSP: Производительность

Ссылка	Задача	Тип графа	# вершин	GTEPS	Число процессоров	Система
Bader, Madduri	BFS	R-MAT	200 M	0.5	40	Cray MTA-2
Checconi et al.	BFS	Graph 500	2^{32}	254	4k узлов / 64k ядер	IBM BG/Q NNSA SC
Graph 500 Nov 2013	BFS	Graph 500	2^{36}	1 427	4k узлов / 64k ядер	IBM BG/Q Mira
Graph 500 Nov 2013	BFS	Graph 500	2^{39}	14 328	32k узлов / 512k ядер	IBM BG/Q Mira
Graph 500 Nov 2013	BFS	Graph 500	2^{40}	15 363	64k узлов / 1M ядер	IBM BG/Q Sequoia
Madduri et al.	SSSP	R-MAT	2^{28}	0.1	40	Cray MTA-2
[sssp-2014]	SSSP	R-MAT	2^{35}	650	4k узлов / 64k ядер	IBM BG/Q Mira
[sssp-2014]	SSSP	R-MAT	2^{38}	3100	32k узлов / 512k ядер	IBM BG/Q Mira

SSSP: Классические алгоритмы

- Dijkstra algorithm
- Bellman-Ford algorithm
- Δ-stepping algorithm (Meyer, Sanders)

SSSP: Обозначения

- $G = (V, E, w)$ – ненаправленный граф
- e – ребро из множества E
- $w(e) > 0$ – целочисленный вес ребра
- $N = |V|$ – число вершин
- $m = |E|$ – число ребер
- `rt` – корневая вершина
- $d^*(v)$ – кратчайшее расстояние от вершины v до корневой вершины `rt`

SSSP: Терминология

- $d(v) \geq d^*(v)$ – “предварительное” расстояние
  - инициализация
    - $d(v) = \infty$
    - $d(\texttt{rt}) = 0$
  - в конце алгоритма
    - $d(v) = d^*(v)$ для всех вершин
- Релаксация ребра
  - пусть $d(u)$ изменилось
  - `Relax`(u, v): $d(v) = \min\{ d(v), d(u) + w(<u, v>) \}$
- Если $d(v) = d^*(v)$, то v – settled-вершина

SSSP: Δ-stepping algorithm

Initialization

Set $d(\texttt{rt}) \leftarrow 0$; for all $v \neq \texttt{rt}$, set $d(v) \leftarrow \infty$.
Set $B_0 \leftarrow \{\texttt{rt}\}$ and $B_\infty \leftarrow V - \{\texttt{rt}\}$.
For $k = 1, 2, \ldots,$, set $B_k \leftarrow \emptyset$.

Δ-Stepping Algorithm

$k \leftarrow 0$.
Loop // Epochs
  ProcessBucket(k)
  Next bucket index : $k \leftarrow \min\{i > k \ : \ B_i \neq \emptyset\}$.
  Terminate the loop, if $k = \infty$.

ProcessBucket(k)

$A \leftarrow B_k$. //active vertices
While $A \neq \emptyset$ //phases
  For each $u \in A$ and for each edge $e = \langle u, v \rangle$
    Do $\texttt{Relax}(u, v)$
  $A' \leftarrow \{x \ : \ d(x)$ changed in the previous step$\}$
  $A \leftarrow B_k \cap A'$

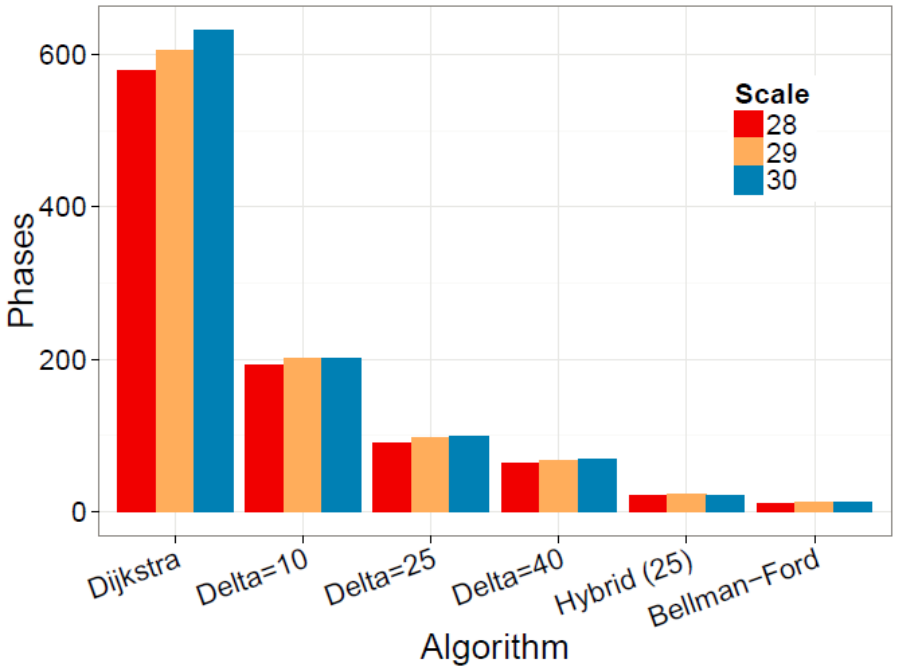
$\texttt{Relax}(u, v)$**:**

Old bucket: $i \leftarrow \lfloor \frac{d(v)}{\Delta} \rfloor$.
$d(v) \leftarrow \min\{d(v), d(u) + w(\langle u, v \rangle)\}$.
New bucket : $j \leftarrow \lfloor \frac{d(v)}{\Delta} \rfloor$.
If $j < i$, move v from B_i to B_j.

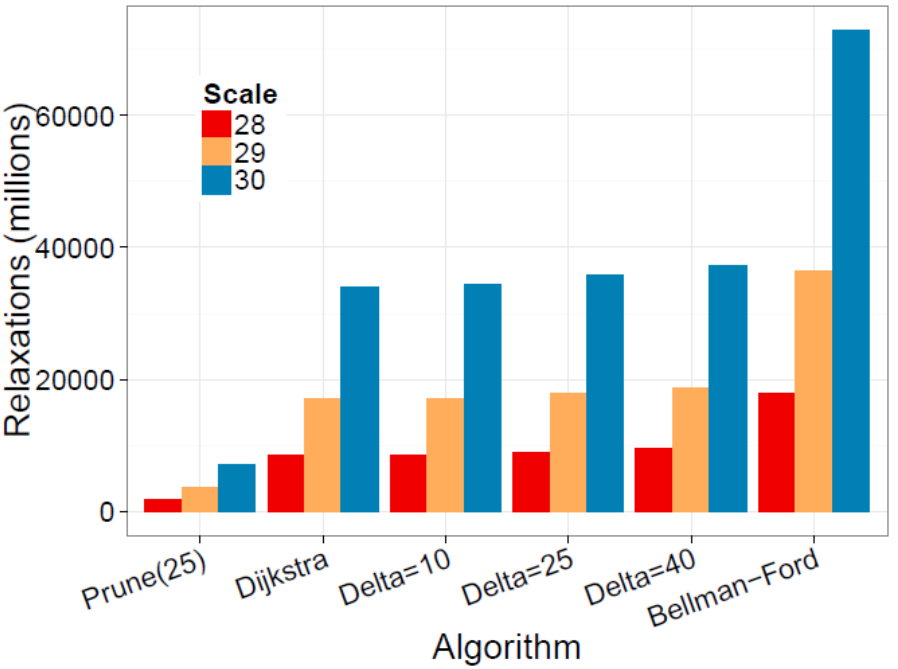
SSSP: Распределенный алгоритм

- Вершины распределены между узлами
- `Relax`(*u*, *v*)
  - узел-владелец вершины *u* посылает значение *d*(*u*) узлу-владельцу вершины *v*

SSSP: сравнение алгоритмов



(a) Number of phases



(b) Number of relaxations

- Число фаз (allreduce)
  - Dijkstra ≥ Δ-stepping ≥ Bellman-Ford
- Число релаксаций (коммуникации, imbalance)
  - Dijkstra ≤ Δ-stepping ≤ Bellman-Ford

SSSP: Особенности алгоритма

- Сокращение числа релаксаций
  - Pruning
    - direction-оптимизация
      - два типа релаксации: push и pull
    - классификация ребер
- Сокращение числа фаз
  - Гибридный подход
    - на начальных шагах – Δ-stepping
    - далее – Bellman-Ford
- Балансировка нагрузки
  - соседи “тяжелых” вершин делятся между узлами

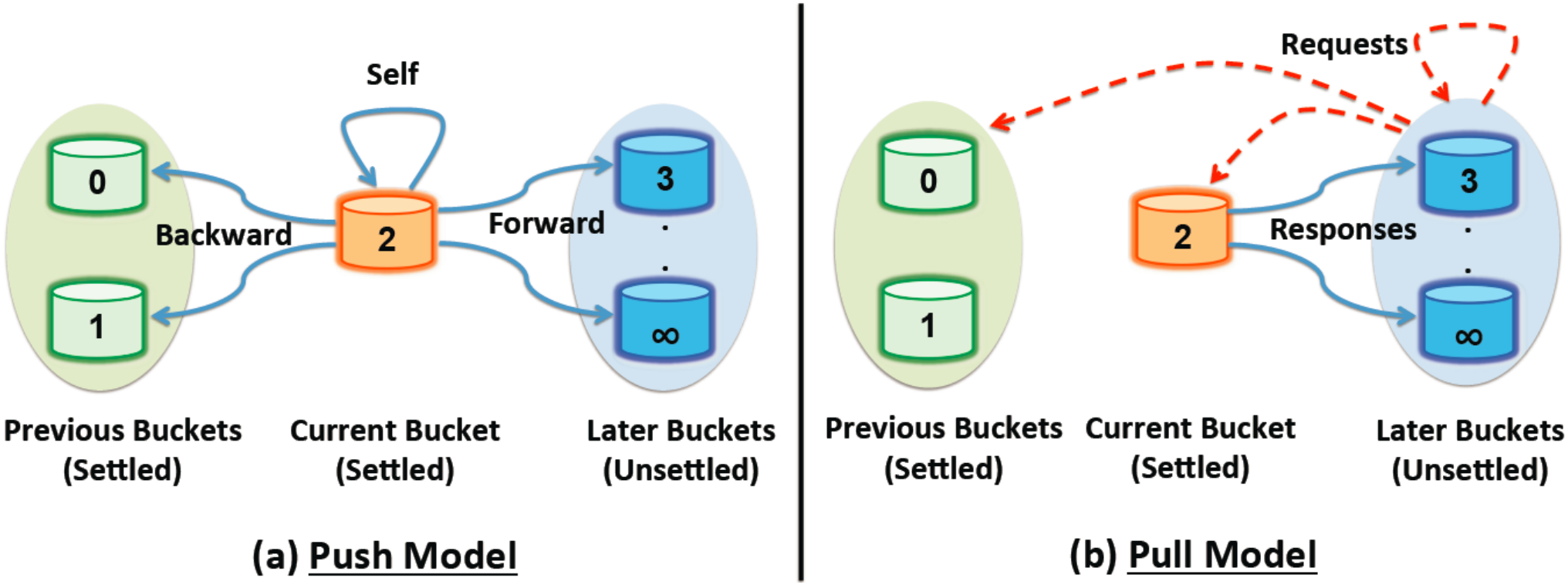
SSSP: Классификация ребер

- Meyer, Sanders
  - $w(e) < \Delta$ – короткие
  - $w(e) \geq \Delta$ – длинные (только в конце фазы)
- Классификация коротких ребер (inner-outer short heuristics, IOS)
  - $d'(v) = d(u) + w(e)$ – входит в B_k?
    - inner short edges – релаксируем немедленно
    - outer short edges – релаксируем вместе с длинными

SSSP: pruning-эвристика

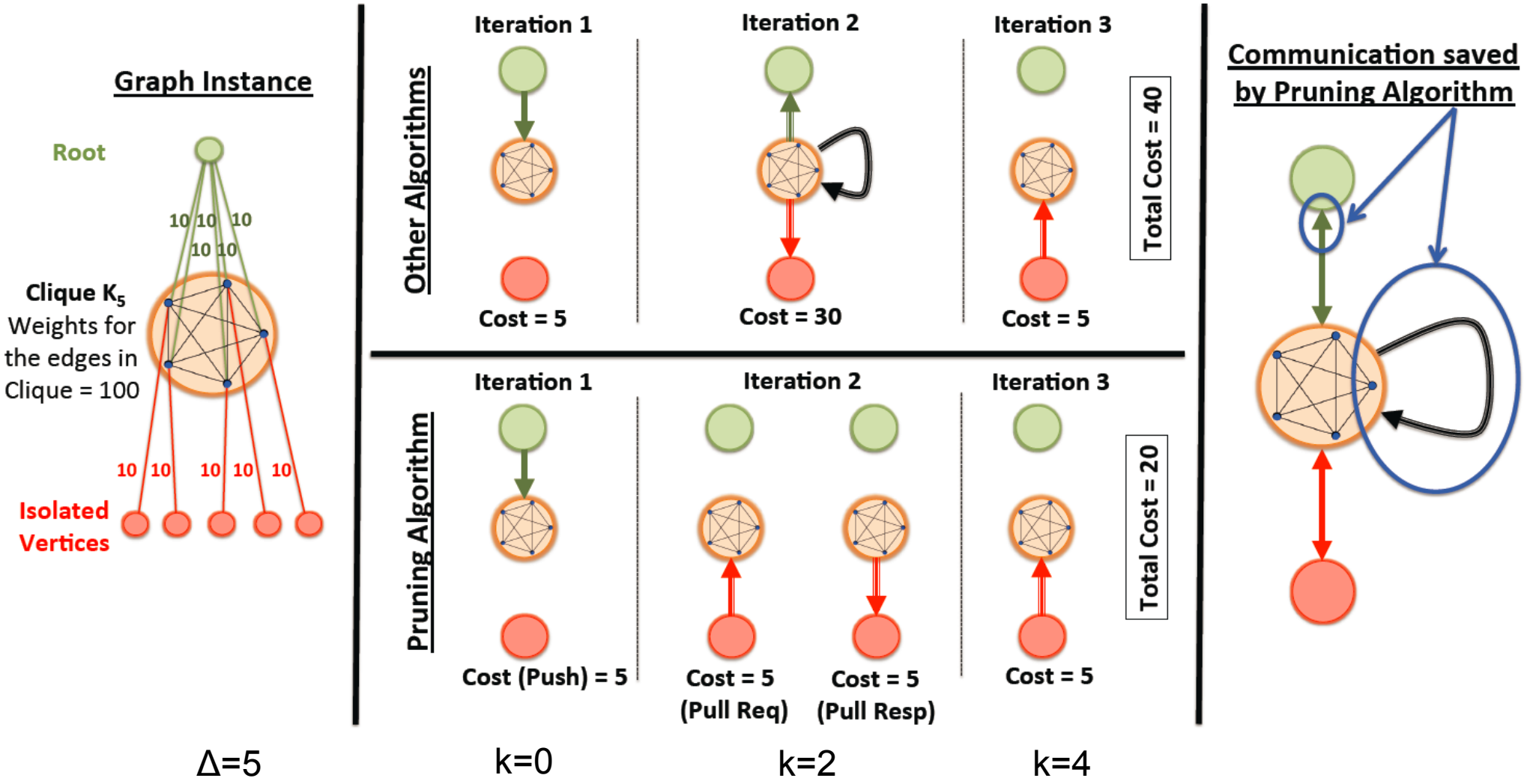
- BFS: direction-оптимизация
- В какой фазе больше релаксаций:
  - фаза коротких ребер?
  - фаза длинных ребер?
  - если $\Delta << w_{max}$, то большинство ребер в $[\Delta, w_{max}]$
- Оптимизируем релаксацию длинных ребер

SSSP: push- и pull-модели



$w(e) < d(v) - k\Delta$

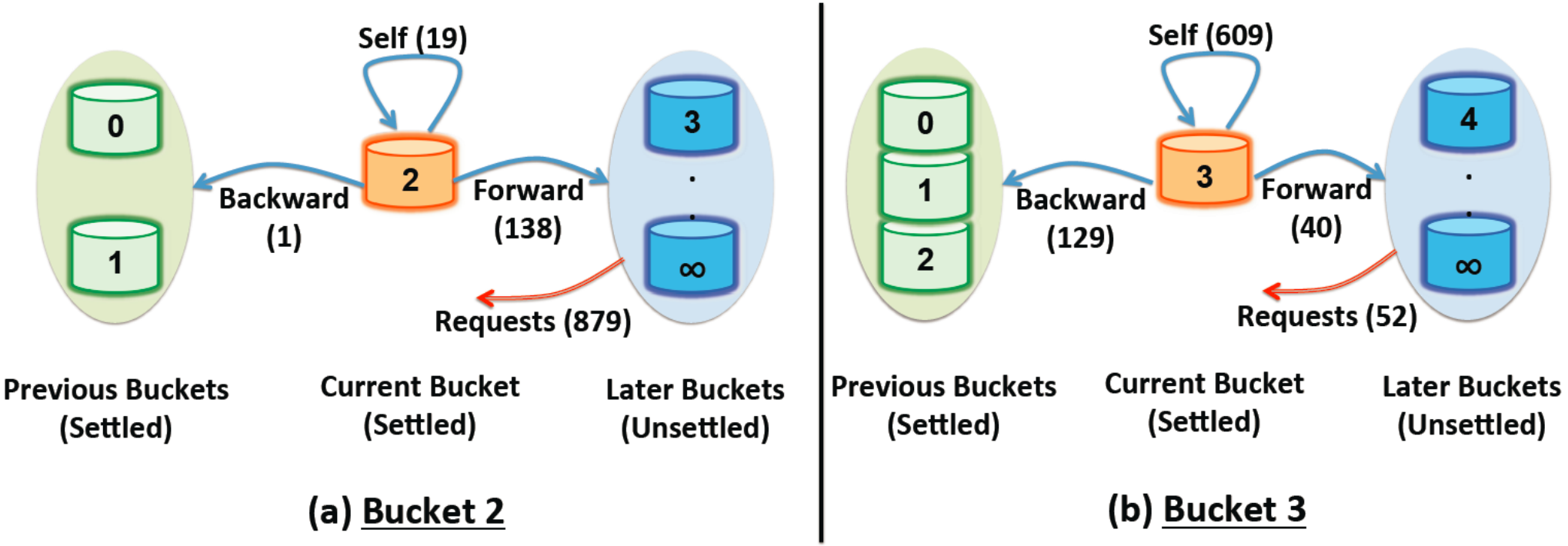
SSSP: Иллюстрация pull-модели



Δ=5 k=0 k=2 k=4

$w(e) < d(v) - k\Delta$

SSSP: Выбор между push и pull



- push-модель – для bucket'ов с вершинами низкой степени
- pull-модель – для bucket'ов с вершинами высокой степени

Заключение

Благодарности

- IBM Research
  - Fabrizio Petrini
  - Fabio Checconi

Заключение

- BFS
  - IBM Blue Gene/Q Sequoia, 64k узлов, 4 млн. нитей
  - scale=40, 15.3 трлн. TEPS
  - 1D-декомпозиция, compressed adjacency list with a coarse index, direction-оптимизация, балансировка нагрузки, сжатие сообщений, многопоточность
- SSSP
  - IBM Blue Gene/Q Mira, 32k узлов, 2 млн. нитей
  - scale=38, 3.1 трлн. TEPS
  - классификация ребер, pruning-эвристика (push-, pull-модели), гибридизация, балансировка нагрузки

Ссылки

- [g500-2014] F. Checconi, F. Petrini “Traversing Trillions of Edges in Real-time: Graph Exploration on Large-scale Parallel Machines”, IPDPS 2014, Phoenix, Arizona, May 19-23, 2014 http://www.odbms.org/wp-content/uploads/2014/05/g500-ipdps14.pdf
- [sssp-2014] V.T. Chakaravarthy, F. Checconi, F. Petrini, Y. Sabharwal “Scalable Single Source Shortest Path Algorithms for Massively Parallel Systems”, IPDPS 2014, Phoenix, Arizona, May 19-23, 2014 http://www.odbms.org/wp-content/uploads/2014/05/sssp-ipdps2014.pdf
- [g500-2012] F. Checconi, F. Petrini, J. Willcock, A. Lumsdaine, A.R. Choudhury, Y. Sabharwal “Breaking the speed and scalability barriers for graph exploration on distributed-memory machines”, SC '12, Salt Lake City, Utah, Nov 10-16, 2012

Дополнительные слайды

IBM Blue Gene/Q Sequioa

- LLNL
- TOP500
  - 3-е место (2013/11)
  - R_{peak} = 20.1 Пфлоп/с
  - R_{max} = 17.2 Пфлоп/с
- 96 стоек (12 x 8)
  - 98 304 выч. узла
  - 1 572 864 ядра
  - 1.5 ПБ памяти

BFS: Проблемы и пути решения

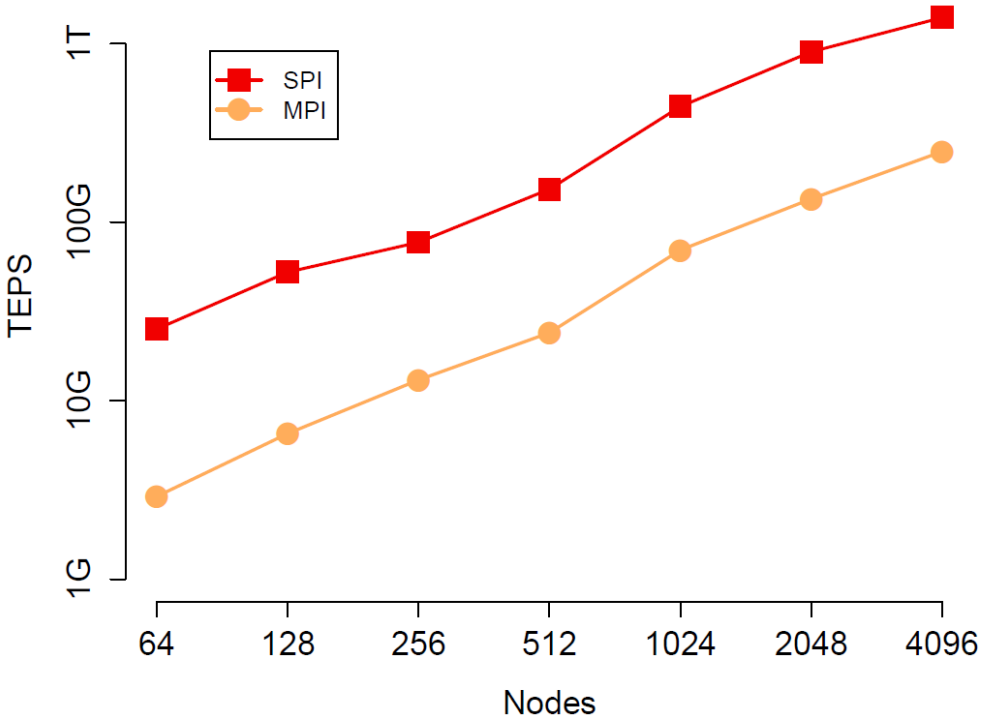
Challenge	Strategy
Algorithms	
Data decomposition	1D/2D
Search pruning	Direction optimization
Metadata	Bitmaps/shortcut lists
Load balancing	Node/thread
Comp/comm equalization	Compression
Implementation	
Irregular communication	Mailboxing/coalescing
Synchronization overhead	Lock-free queues
QoS	Private queues
Global decisions	Async collectives

BFS: Сжатие сообщений

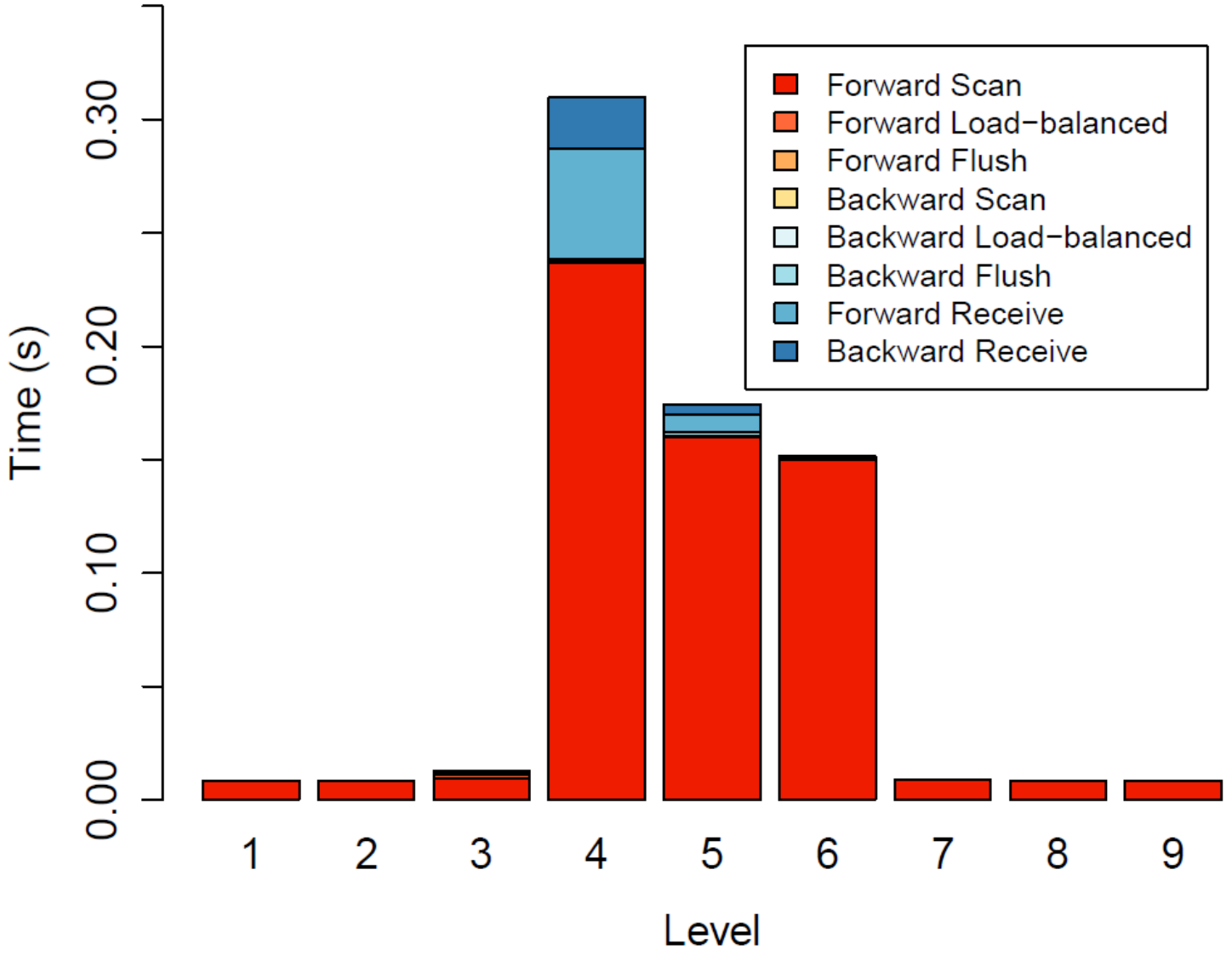
- Пакет
  - заголовок (64 бита)
    - идентификатор src-узла (16 бит)
    - содержательная нагрузка пакета в байтах (24 бита)
    - базовая вершина (24 бит)
  - вершина (48 бит)
    - младшие биты src-вершины (24 бита)
    - младшие биты dst-вершины (24 бита)
  - reduced-формат (32 бита)
    - младшие биты src-вершины (24 бита)
    - разница с предыдущей / базовой dst-вершиной (7 бит)

BFS: SPI vs MPI

- SPI:
  - Blue Gene/Q System Programming Interface
  - асинхронный
- MPI:
  - bulk-synchronous
  - две фазы (подсчет, обмен)
- Direction-оптимизация
- max scale per node:
  - SPI: 24
  - MPI: 23



BFS: Разбиение по времени



1024 узла, scale=34

IBM Blue Gene/Q Mira

- ANL
- TOP500
  - 5-е место (2013/11)
  - R_{peak} = 10.1 Пфлоп/с
  - R_{max} = 8.59 Пфлоп/с
- 48 стоек
  - 49 152 выч. узла
  - 786 432 ядра
  - 0.75 ПБ памяти

Правовая информация

IBM, логотип IBM, BladeCenter, System Storage и System x являются товарными знаками International Business Machines Corporation в США и/или других странах. Полный список товарных знаков компании IBM смотрите на узле Web: www.ibm.com/legal/copytrade.shtml.

Названия других компаний, продуктов и услуг могут являться товарными знаками или знаками обслуживания других компаний.

Упоминание в этой публикации продуктов или услуг корпорации IBM не означает, что IBM предполагает предоставлять их во всех странах, в которых осуществляет свою деятельность, информация о предоставлении продуктов или услуг может быть изменена без уведомления. За самой свежей информацией о продуктах и услугах компании IBM, предоставляемых в Вашем регионе, следует обращаться в ближайшее торговое представительство IBM или к авторизованным бизнес-партнерам.

Все заявления относительно намерений и перспективных планов IBM могут быть изменены без уведомления.

Информация о продуктах третьих фирм получена от производителей этих продуктов или из опубликованных анонсов указанных продуктов. IBM не тестировала эти продукты и не может подтвердить производительность, совместимость, или любые другие заявления относительно продуктов третьих фирм. Вопросы о возможностях продуктов третьих фирм следует адресовать поставщику этих продуктов.

Информация может содержать технические неточности или типографические ошибки. В представленную в публикации информацию могут вноситься изменения, эти изменения будут включаться в новые редакции данной публикации. IBM может вносить изменения в рассматриваемые в данной публикации продукты или услуги в любое время без уведомления.

Любые ссылки на узлы Web третьих фирм приведены только для удобства и никоим образом не служат поддержкой этим узлам Web. Материалы на указанных узлах Web не являются частью материалов для данного продукта IBM.